# Unreliable Test Infrastructures in Automotive Testing Setups

Claudius Jordan
Philipp Foth
Alexander Pretschner
Technical University of Munich
Munich, Germany

Matthias Fruth
TraceTronic GmbH
Dresden, Germany

## ABSTRACT

During system testing of automotive electrical control units various reasons can lead to invalid test failures, e.g., non-responding components, faulty simulation models, faulty test case implementations, or hardware or software misconfigurations. To determine whether a test failure is invalid and what the underlying cause was, the test executions have to be analyzed manually, which is tedious and therefore costly. In this work, we report the magnitude of the problem of invalid test failures with four system testing projects from the automotive domain. We find that up to 91% of failed test executions are considered invalid. An oftentimes overlooked challenge are unreliable test infrastructures which deteriorate the validity of the test runs. In the studied projects already between 27% and 53% of failed test executions are linked to unreliable test infrastructures.

**ACM Reference Format:**
Claudius Jordan, Philipp Foth, Alexander Pretschner, and Matthias Fruth. 2022. Unreliable Test Infrastructures in Automotive Testing Setups . In *44th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP '22), May 21–29, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3510457.3513069

## 1 MOTIVATION

Testing modern automotive systems is a major challenge and the required test infrastructures are becoming increasingly complex due to the complex distributed functionality and variant diversity of modern vehicles. This growing complexity brings its own challenges, including the error-proneness of such test infrastructures. In practice, hardware-in-the-loop test beds (HiLs), which are used for system testing, suffer from sporadic malfunctioning [1–4]. Because produced test results cannot be regarded reliable, the actual goal of validation & verification (V&V) – identifying faults in the system-under-test (SUT) – is impeded. In fact, the overall test automation value is diminished if many invalid results are produced. The problem becomes clearer when realizing that each failed test execution needs to be analyzed by test experts whose time is a scarce and costly resource.

In this study we approach the following questions: *How many invalid test failures are produced during automated test execution in automotive system testing? How many of those invalid test failures can be ascribed to unreliable test infrastructures?*

In the following, we describe how we collect the data to answer those questions (§2), discuss the insights we gain from the datasets (§3) and conclude with our plans for continuing this study and eventually addressing the problem (§4).

*ICSE-SEIP '22, May 21–29, 2022, Pittsburgh, PA, USA*

## 2 DATA COLLECTION

The subjects of study are four ongoing V&V projects from our industry partner, a leading company for test automation in the automotive domain. In the examined projects, the SUT are electrical control units (ECUs) responsible for system-level customer-functions. For their fulfillment various car sub-systems such as the power train and the on-board supply system are involved, e.g., automatic motor stop at traffic lights. In general, test case executions (TCEs) with a result other than *passed* are reviewed and assessed. For this, the test experts employ the test report database TEST-GUIDE[1] which we use to collect the data for this study. The assessment consists of diagnosing the failure cause and documenting it in a review in said test report database. Each review usually contains, among other things, an assessment comment and ideally also a defect class. Here, defect classes represent categories of failure causes and in each project a project-specific list of defect classes is specified out of which the reviewers select one. Hence, to answer above questions we can directly use the project-defined defect classes.

For comparability, we collect data for the same time period for all four projects. As in two projects, defect classes have been adopted rather recently, we use data from a few months only even though the projects already run for several years and, hence, more assessment data is available – only without assigned defect classes. In addition, for various reasons, up to half of the assessed TCEs have no defect class assigned. In both cases, one could use other available information such as links to tickets in corresponding issue management systems or the handwritten assessment comments to assign these TCEs to defect classes. However, we do not perform any additional labeling to avoid introducing bias.

## 3 FINDINGS

We group the defect classes into three broader failure cause categories: We distinguish between code defects (*CD*), test infrastructure issues (*TI*) and other causes for failures (*OC*). Hereby, only the TCEs of category *CD* represent the desired *valid* test failures as in this case actual faults in the SUT lead to the failed TCE. *TI* includes hardware and software failures, e.g., defective components, faulty simulation models, and unstable software tools. *OC* comprises rather operator-dependent issues such as configuration mistakes, test case implementation errors, incorrect or outdated test specifications. Note, *TI* and *OC* also differ by the fact that for more severe, oftentimes recurring test infrastructure issues (*TI*) subsequent actions are provoked for the responsible infrastructure

[1] https://www.tracetronic.com/products/test-guide

**Table 1: Overview of the studied projects**

| | | Project | | | |
|---|---|---|---|---|---|
| | | A | B | C | D |
| # test cases | | 0.2k | 0.5k | 2k | 1k |
| # assessed TCEs | | 3k | 11k | 30k | 13k |
| # TCEs with defect class | | 2k | 11k | 15k | 6k |
| Category portion (%) | CD[2] | 26 | 10 | 11 | 9 |
| | TI | 53 | 44 | 32 | 27 |
| | OC | 21 | 46 | 57 | 64 |

providers to fix, while for other causes (*OC*) oftentimes a tester can resolve the issue and then rerun the test. Together, *OC* and *TI* account for undesired *invalid* test failures.

In Tab. 1, we list approximations of the number of test cases, the number of assessed TCEs, and the number of those with a defect class assigned. In addition, we compile the portions of the defect class categories relative to the number of TCEs with defect classes.

One of the projects which only recently started to assign defect classes to their assessments is project A. In terms of test cases, it is the smallest of the projects. With 26% of TCEs being linked to code defects, it has by far the highest portion. With 53%, it also seems to suffer the most from *TI* issues.

Project B began to use TEST-GUIDE only recently for assessing TCEs. Their motivation is to gain a better view on the testing process quality. This explains the outstanding situation of practically all assessed TCEs having defect classes assigned. Here, around 10% of failed TCEs are linked to faults in the SUT. The remaining 90% of invalid test failures are evenly distributed among *TI* and *OC*.

The largest project in terms of test executions is project C. The portion of *CD* is similar to project B while the portion of *TI* is apparently lower. However, it is noticeable that especially for TCEs of category *OC* the same reviews are repeated many times indicating that the same issues keep recurring for many TCEs. In fact, only around 500 unique reviews have been assigned to the TCEs of category *OC* while for those of category *TI* there are almost 400 despite the substantial difference in absolute numbers of assessments (57% compared to 32%). Moreover, the experts told us that many times no defect class is assigned if a blunt rerun of a previously failed test passes without any operator intervention, as no in-depth analysis of the underlying failure cause is performed in such cases. This suggests that a substantial amount of sporadic *TI* issues might be hidden in the unlabeled TCEs, i.e., those for which no defect class has been assigned.

In project D, for all issues 'under investigation' no defect classes usable for our study are assigned. In fact, in those cases the review comments rather reflect the actual ongoing analysis process. Unfortunately, the defect classes are not updated once the cause has been identified. Like for project C, in the discussions the experts suggested that a substantial portion of the unlabeled TCEs could be considered as *TI*. The reason is that unlike the other projects, they explicitly do not categorize unspecific sporadic problems.

[2]Recall that in the test report database there are many assessed failed TCEs without defect class assignments. The test experts do not expect (many) code defects among them. Hence, the numbers presented in Tab. 1 for category $CD$ can be considered rather as upper bound for the studied projects.

## 4 CONCLUSION

We identify test infrastructure unreliability as a challenge that is impacting system testing with hardware-in-the-loop test benches. In the studied projects invalid test failures amount up to 91% of all failed test executions while test infrastructure issues already account for 27% to 53%. This preliminary study is intended to raise awareness of this challenge, for which, to our knowledge, we are the first to provide numbers on the extent of the problem. A major limitation is that the projects are from only one company and are therefore not representative of system testing across the automotive industry. In addition, defect classes are determined by project stakeholders and, as far as we know, there are no best practices to date to ensure consistency. Consequently, the mapping of the project-specific defect classes was decided in consultation with the experts from each project. Overall, we believe that the numbers are consistent with the experience of many test practitioners. Clearly, the challenges posed by invalid test failures and, in particular, unreliable test infrastructures are significant and need to be addressed.

In general, we see two ways to improve the situation for practitioners. First, the causes of unreliable test infrastructures could be addressed. There are several levers that can be applied here, both at the organizational and technical level. On the other hand, we see the potential to support test experts in the review process and thus coping with the present test infrastructure unreliability. Therefore, we will investigate how to automatically determine the failure causes (i.e., the defect classes) of failed TCEs. In addition, we will extend the study to further investigate what factors influence the occurrence of invalid test failures.

## ACKNOWLEDGMENTS

This work was supported by the German Federal Ministry for Economic Affairs and Energy (BMWi).

## REFERENCES

[1] Roozbeh Bakhshi, Surya Kunche, and Michael Pecht. 2014. Intermittent failures in hardware and software. *J. Electron. Packag. Trans. ASME* 136, 1 (2014), 011014. https://doi.org/10.1115/1.4026639

[2] Isabel Evans, Chris Porter, Mark Micallef, and Julian Harty. 2020. Stuck in Limbo with Magical Solutions: The Testers' Lived Experiences of Tools and Automation. In *Proc. 15th Int. Jt. Conf. Comput. Vision, Imaging Comput. Graph. Theory Appl.* SCITEPRESS - Science and Technology Publications, Valletta, Malta, 195–202. https://doi.org/10.5220/0009091801950202

[3] Per Erik Strandberg, Thomas J. Ostrand, Elaine J. Weyuker, Wasif Afzal, and Daniel Sundmark. 2020. Intermittently failing tests in the embedded systems domain. In *Proc. 29th ACM SIGSOFT Int. Symp. Softw. Test. Anal.* ACM, New York, NY, USA, 337–348. https://doi.org/10.1145/3395363.3397359 arXiv:2005.06826

[4] Kristian Wiklund, Sigrid Eldh, Daniel Sundmark, and Kristina Lundqvist. 2017. Impediments for software test automation: A systematic literature review. *Softw. Test. Verif. Reliab.* 27, 8 (2017), 1–20. https://doi.org/10.1002/stvr.1639